SO2 (14/01/2025)

Final de laboratorio

Crea un fichero que se llame "respuestas.txt" donde escribirás las respuestas para los apartados de los ejercicios del control. Indica para cada respuesta, el número de ejercicio y el numero de apartado (por ejemplo, 1.a, 1.b, ...).

Importante: para cada uno de los ejercicios tienes que partir de la versión de Zeos original que te hemos suministrado.

1. (2 puntos) PF

Reprograma la excepción de fallo de página para que cuando un proceso genere un fallo de página se muestre por pantalla:

```
Process XXX has generated a page fault at YYY
```

Donde XXX corresponde al PID del proceso y YYY corresponde a la instrucción que ha generado la excepción. A continuación se matará al proceso y se pasará a ejecutar otro proceso disponible.

2. (2 puntos) Copy others memory

Queremos implementar la siguiente llamada a sistema:

```
int copy_user_mem(int pid, char* start,
                  char* dest, int size);
```

Esta función copia *size* bytes de memoria del proceso pid desde la dirección *start* en la dirección *dest* del proceso actual. Esto nos permite realizar un código como el siguiente:

```
1.  int value = 666;
2.  p = fork();
3.  if (p==0) { value  = 42; }
4.  else if (p > 0) {
5.    int local;
6.    copy_user_mem(p, &value, &local, sizeof(int));
7.  }
```

Suponiendo que esta función ya estuviera implementada:

a) ¿Qué valor contendrá el proceso padre en la variable *local* después de ejecutar la línea 6?

Indica el código necesario para:

b) Calcular la página que contiene la dirección *start.*

c) Mapear la zona de memoria *[start, start+size)* del proceso pid en las últimas direcciones del espacio de direcciones del proceso actual. Puedes suponer que dispones de una función *find_task_by_pid* que dado un pid retorna un puntero a su PCB y que siempre se va a poder hacer el mapeo.

d) Copiar con una única llamada a *copy_data* desde la zona mapeada a la zona de destino.

e) Desmapear la zona de memoria.

3. (6 puntos) Call me maybe

Queremos añadir a nuestro ZeOS una funcionalidad para que se ejecute una función al cabo de un tiempo. La sintaxis es :

```
int timeout(int tics, void (*f)(void));
```

Esta llamada programa la ejecución de la rutina ´f´ por el proceso actual cuando hayan pasado ´tics´ ticks de reloj. Esta llamada devuelve el numero de ticks que faltaban para que saltara el siguiente timeout o 0 si no estaba programado.
Cada tick de reloj se debe comprobar si hay procesos pendientes de timeout y si se les ha acabado el tiempo, marcarlos. Estos procesos marcados, al volver a modo usuario, pasarán a ejecutar la funcion 'f' y luego continuarán su ejecución en el punto donde estuvieran. Para ello, **justo antes de que un proceso restaure su contexto hardware para volver a modo usuario,** hay que comprobar si el proceso está marcado para ejecutar la rutina de timeout y en ese caso:

1) modificar el contexto hardware para que ejecute la rutina ´f´,
2) modificar la pila de usuario añadiendo en su cima la dirección a la que volver cuando termine la rutina ´f´ (la que hubiera usado el proceso al entrar en sistema) simulando un ´call f´ desde ese punto y
3) modificar el contexto hardware acorde a la modificación del punto anterior.

Esta llamada a sistema tiene que usar el identificador de servicio 7 para realizar la entrada a sistema. Los parámetros se pasarán por registro. La programación de un timeout no debe heredarse a sus hijos.
La única restricción de la rutina ´f´ es que al entrar y salir de la función tiene que llamar a las funciones SAVE_REGS y RESTORE_REGS respectivamente para guardar y restaurar el contenido de todos los registros (podéis suponer que ya estan implementadas). Si no se llaman y la función no acaba con un ´exit´ el comportamiento es indeterminado.
La solución tiene que ser genérica y funcionar de forma eficiente para cualquier número de procesos.
Se pide:

a) (0,5 puntos) Implementa el código del wrapper de la llamada *timeout.*
b) (0,5 puntos) Indica los cambios necesarios en los handlers existentes.
c) (0,5 puntos) Indica qué estructuras de datos se tienen que añadir y/o modificar. Añade el código necesario para inicializarlas.
d) (1 punto) Implementa el código de la rutina *sys_timeout*.
e) (1 punto) Implementa los cambios necesarios en la rutina de reloj para detectar el timeout.
f) (1,5 puntos) Indica el código necesario para detectar si un proceso está marcado y ejecutar su rutina de timeout.
g) (0,5 puntos) ¿Es necesario modificar alguna otra llamada a sistema o parte del sistema para implementar por completo esta funcionalidad? Si es así implementa los cambios necesarios.

Entrega

Sube al Racó el fichero “respuestas.txt” junto con el código que hayas creado en cada ejercicio.

Para entregar el código utiliza:

```
> tar zcfv examen.tar.gz ejercicio1 ejercicio2 ejercicio3 respuestas.txt
```